

AD-A091 244

AIR FORCE WEAPONS LAB KIRTLAND AFB NM

F/6 9/2

A SOLUTION TO THE LARGE CODE PROBLEM ON THE AFWL SCIENTIFIC COM--ETC(U)

SEP 80 R W BERRY

UNCLASSIFIED

AFWL-TR-80-20

SBIE-AD-E200 5A4

NI

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

1-0

END

DATE

FILED

1-8

DTIC

AFWL-TR-80-20

② LEVEL III
SC

AP-E200584

AFWL-TR-
80-20

A SOLUTION TO THE LARGE CODE PROBLEM ON THE AFWL SCIENTIFIC COMPUTERS

Maj R. W. Berry

September 1980

Final Report

Approved for public release; distribution unlimited.

DTIC
ELECTE
S NOV 5 1980 D
B

AIR FORCE WEAPONS LABORATORY
Air Force Systems Command
Kirtland Air Force Base, NM 87117

80

60

AD A091244

DDC FILE COPY



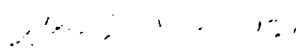
This final report was prepared by the Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico, under Job Order 24440101. Major Robert W. Berry (ADX) was the Laboratory Project Officer-in-Charge.

When US Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been authored by an employee of the United States Government. Accordingly, the United States Government retains a nonexclusive royalty-free license to publish or reproduce the material contained herein, or allow others to do so, for the United States Government purposes.

This report has been reviewed by the Public Affairs Office and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


ROBERT W. BERRY
Major, USAF
Project Officer


LEONARD STANS
Lt Colonel, USAF
Chief, Advanced Systems Projects Br

FOR THE DIRECTOR


DAVID E. MCINTYRE
Chief, Computational Division

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFWL-TR-80-20	2. GOVT ACCESSION NO. AD-A091244	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A SOLUTION TO THE LARGE CODE PROBLEM ON THE AFWL SCIENTIFIC COMPUTERS	5. TYPE OF REPORT & PERIOD COVERED Final Report	
7. AUTHOR(s) Maj R. W. Berry	6. PERFORMING ORG. REPORT NUMBER	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Weapons Laboratory (ADX) Kirtland AFB, NM 87117	8. CONTRACT OR GRANT NUMBER(s)	
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Weapons Laboratory (ADX) Kirtland AFB NM 87117	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62601F 24440101	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	12. REPORT DATE September 1980	
	13. NUMBER OF PAGES 24	
	15. SECURITY CLASS. (of this report) UNCLASSIFIED	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Minicomputer Large Code Scientific Computing Prime 750	IBM 4341 Vector Processing Array Processor Scientific Computer	Scientific Programming VAX-11/780 Array Processing
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
<p>This report addresses the issue of executing large and long running codes on the Air Force Weapons Laboratory large-scale scientific computers, and the problem of obtaining results in a reasonable amount of time with minimum cost. Impact on other users is discussed as well as inefficiencies resulting from the diverse job mix on the large machines. The ALFA code is used to present cost information relating to the current mode of operation.</p> <p style="text-align: right;">Continued</p>		

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Item 20 continued.

An alternative to this mode of operation is offered. A 32-bit minicomputer system which uses a virtual memory addressing scheme and incorporates fast, large capacity disk storage, suitable I/O equipment, sufficient high speed main memory, and a powerful attached array processor is recommended. This system would be used in dedicated mode by one user at a time. The cost advantages of this approach are presented, and continuing actions on this project are recommended.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

CONTENTS

INTRODUCTION	3
CANDIDATE CODES	6
MINICOMPUTER REQUIREMENTS	8
SURVEY OF INDUSTRY MINICOMPUTER HARDWARE	10
CODE CONVERSION REQUIREMENTS	17
ECONOMIC ANALYSIS	19
BENCHMARK SET	22
CONCLUSIONS AND RECOMMENDATIONS	23
ABBREVIATIONS AND SYMBOLS	24

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

INTRODUCTION

PROBLEM

There are currently several large and long running programs being processed on the AFWL scientific computers (CDC 6600, CYBER 176). They are large in terms of the main memory, extended main memory (ECS/LCM), or disk storage requirements, and long running in terms of central processor or total real-time required to execute. These programs must compete for computing resources along with all other jobs which run on the scientific machines. The other jobs include small interactive programs, debug jobs, one-time special purpose jobs, accounting jobs, and others which individually do not require a significant percentage of the computer resources, but together comprise a substantial portion of the total workload. The result of this competition is that the large jobs are run only during nonpeak workload hours or the other work receives less than satisfactory response and turnaround. Usually a combination of these two results occurs wherein special times are scheduled to run the large codes or they are run along with the other jobs. This results in degraded response for all jobs and inefficient machine use.

In addition to the programs which are currently being run, some can be identified for which requirements exist but which are not being run due to lack of sufficient computer resources. In some cases these are the 3-dimensional (3-D) versions of currently running 2-D codes. In all cases they are not being run due to lack of sufficient memory or CP power.

PROPOSED SOLUTION

Many significant advances in minicomputer technology have been achieved during the past several years. These include the development of fast memories; high bandwidth memory and I/O busses; cache memories to provide faster effective memory access; powerful floating point processors, including several general purpose array processors; and large, fast disk storage devices. It is possible that a suitably equipped minicomputer system could be configured which would support the processing and I/O requirements of many of the large codes which are a problem in the current large machine environment.

The minicomputer approach has several advantages. Run in a dedicated (non-multi-tasking) manner, the minicomputer could be operated under the direct control of one group or project for an assigned period of time. During this time, large blocks could be reserved for one or more problems. Overhead due to

scheduling and job swapping would be eliminated. A code which receives one day turnaround for two CP hours of 6600 processing could conceivably be turned around in 2 or 3 hours on the minicomputer. A problem which runs for 40 hours of CP time could be completed in days rather than weeks. In general, although the raw computing power of the minicomputer might be less than the 6600, turnaround times for some codes could be dramatically reduced.

SCOPE

The problem of determining the suitability and economic advantages of using minicomputers to support large computer codes involves identifying the main areas of concern and performing systematic investigations of those areas. In addition, some benchmark codes must be developed to measure the performance of selected minicomputers in comparison with the performance on the large machines. If it is determined that the minicomputer approach is justified, then code conversion and development efforts must be undertaken.

A list of the main areas of concern which have been identified follows. In general, they are listed in the order in which they should be addressed. However, in some cases some overlap can occur. For example, the benchmark set could be developed during the selection and procurement process.

- Identify candidate codes.
- Determine requirements of a minicomputer.
- Accomplish an industry survey of minicomputer hardware.
- Determine code conversion requirements.
- Perform an economic analysis.
- Develop a benchmark set.
- Select and procure a minicomputer system.
- Perform the code conversion and documentation.
- Document results of the effort.

BACKGROUND

During the preparation of this report, the first five areas were investigated in some detail. This included interviewing various key people in the AFWL computer user community to identify candidate codes. Some listings were obtained and studied to determine the amount of effort involved in the conversion process, and general minicomputer requirements were established. Minicomputer hardware which could support the effort was narrowed to three vendors, and the economic analysis

AFWL-TR-80-20

was performed using those vendors. The five areas which have been investigated and the four remaining task areas are discussed in greater detail throughout the remainder of this document.

CANDIDATE CODES

GENERAL

The first step in the effort to identify candidate codes was to identify existing and proposed codes which satisfy the criteria given in the Introduction section, and for which there exist reasonably long term requirements. Reasonably long term is loosely defined as a period of time which would economically justify the code conversion and documentation efforts required, and depends largely upon the extent of the conversion effort and the projected amount of use of the code. The following criteria served as guidelines in identifying prospective codes. One or more were used to determine whether or not a particular code was a candidate to be investigated further. The criteria are stated in terms of current or projected 6600/176 requirements.

- Requires more than 1 hour of 6600 CP time.
- Runs longer than 1 hour on a dedicated machine.
- Typically requires more than 1 day to turn around.
- Requires more than 100 K (octal) words of main memory.
- Requires more than 200 K (octal) words of extended main memory.
- Transfers more than ten million words of I/O.
- Requires more than four million words of disk storage.
- Is classified.

At this time, five codes have been identified as candidates for conversion to the minicomputer environment. These are discussed in the following paragraphs. Table 1 summarizes the important characteristics of these codes.

ALFA

ALFA is a finite difference code being run by AFWL/ARAO personnel in support of the laser R&D effort. It requires a small preprocessor called DYNDIM. Both are entirely FORTRAN programs. ALFA requires up to a half million words of memory and can run for as long as 2 hours on the CDC 6600. Its use is anticipated for at least two more years.

APACHE

This is another finite difference code with characteristics similar to ALFA. The primary physical differences are that APACHE requires up to one million words of memory and could run as long as 50 CP hours on a problem on the CDC 6600.

TABLE 1. CANDIDATE CODES

Name	Memory Rqmts*	CY176/CP Time	Precision/Range
ALFA	165 K/635 K	5 min - 2 hrs	3 digits/ 10^{+21-20}
APACHE	256 K/985 K	2 - 50 hrs	3 digits/ $10^{+20, -20}$
HULL	256 K/1959 K**	30 - 40 hrs	12 digits/ $10^{+30, -30}$
NASTRAN	96 K/734 K**	Minutes	12 digits/ $10^{+30, -30}$
QUANTA	256 K/985 K	Minutes	6 digits/ $10^{+4, -4}$

*Memory requirements are listed in decimal numbers as total SSM + LCM CY176 60-bit words/total minicomputer 8-bit bytes. The minicomputer requirement assumes that 95 percent of the CY176 memory requirements consists of floating point data.

**Requires double precision floating point numbers.

HULL

The HULL code is currently being investigated for conversion to run on the GRAY I at the AFWL. One version or another of HULL has been running at the AFWL for several years, and its use is expected to continue. A typical HULL problem can require up to 40 CP hours of CYBER 176 time to complete and can use a million words of memory. It is expected that HULL will be one of the most difficult codes to convert for this effort.

NASTRAN

NASTRAN is a structural analysis code which is used rather heavily at AFWL. However, the current version is supported by a contractor and is proprietary. It is probably not feasible to accomplish the conversion using in-house resources. At least two minicomputer versions exist at this time. A version for the PRIME is available through Schaeffer analysis in Mt Vernon, New Hampshire, and a VAX version is available through MacNeal-Schwendler in Los Angeles.

QUANTA

QUANTA is a war gaming simulation code concerned with weapons allocation and optimization. Although the code requires only minutes of CPU time, the production version is expected to require a million words of memory and will be classified. These characteristics together with the fact that it is written entirely in FORTRAN make QUANTA a good candidate for the minicomputer environment.

MINICOMPUTER REQUIREMENTS

GENERAL

After identifying candidate codes, the baseline requirements of a minicomputer were formulated. This was done using the following general guidelines:

- Overlaying requirements should be kept to a minimum. Therefore, a large address space is necessary.
- Floating point range and accuracy requirements must be satisfied.
- Some tradeoffs can be made in the area of required hardware performance versus improved turnaround in a dedicated or semidedicated environment.

HARDWARE CAPACITY AND PERFORMANCE CONSIDERATIONS

The following characteristics relating to capacity or performance of various hardware features were considered:

- Main memory capacity.
- Main memory addressability.
- Main memory bandwidth.
- Mass storage capacity.
- Mass storage transfer rate.
- I/O bus bandwidth.
- CP timing (typical or average add, multiply, and divide times).
- Floating point accuracy (significant digits).
- Range of exponent.

GENERAL HARDWARE REQUIREMENTS

The following are general requirements which the hardware must support:

- Large capacity (100 M bytes) removable disk packs (3330/844 technology) must be available.
- The hardware must support direct-memory-access I/O.

SOFTWARE REQUIREMENTS

The following requirements must be satisfied by the supplied software:

- The operating system must support multitasking and foreground/background modes of operation.
- Sequential and direct access (random) files must be supported.

- The file system must contain privacy and protection features.
- As a minimum, an optimizing ANSI FORTRAN compiler must be available. A PASCAL compiler is desirable.
- A complete mathematical library must be provided, including single and double precision trigonometric routines.
- File editing and sort utilities must be provided.
- ASCII nine track tape format must be supported.
- Graphics software must be available.

MEMORY REQUIREMENTS

Given a minicomputer system which is otherwise suited to the application for which it is being used, lack of sufficient addressable memory is an annoying and restrictive problem with most machines. In fact, this problem is not confined to minicomputers—the large, scientific computers in use today also suffer from lack of memory. Several minicomputers are available today which have solved this problem, or at least eased it to a large extent. The virtual memory systems offer enough addressable memory to satisfy even the most voracious memory users.

Most minicomputer systems still restrict the programmer to a 64 K byte address space. Due to the magnitude of the programming and conversion problems which can be caused by this restriction, it was decided that no system so limited would be considered a viable candidate system for this application. Rather, a virtual memory system will be required. In addition, a decision was made to consider only 32-bit machines. This would generally provide greater speed over the 16-bit systems and more compatibility and transportability over the 22-bit (HP3000-III) and 24-bit (Harris 570) machines. These restrictions limited the choices to the three vendors which are discussed in the following section.

SURVEY OF INDUSTRY MINICOMPUTER HARDWARE

OFF-THE-SHELF SYSTEMS

After the requirements for a minicomputer system were determined, an industry survey was accomplished to decide if those requirements could be satisfied. Initially, the list of possible vendors was lengthy and included such vendors as Data General, Hewlett-Packard, Honeywell, Modcomp, Harris and Sperry-Univac. However, the virtual memory and 32-bit architecture constraints reduced the list to Prime, IBM and Digital Equipment Corporation.

Table 2 lists some of the important characteristics of each vendor's candidate system. Note that the table does not include any mention of mass storage devices. Each of the vendors supports a disk system using the IBM 3330 technology. This is a 200M-300M byte disk storage unit with an average access time of 30 ns and a maximum transfer rate of 1.2 MB per second.

PRIME. PRIME Computer, Inc., of Farmingham, Mass., offers a wide range of performance in its minicomputer line, from the single user PRIME 100 and 200 to the multiuser 16-bit PRIME, 300, 350, 400 and 500 to the 32-bit PRIME 550, 650 and 750. The PRIME 750 is in the VAX-11/780 and IBM 4341 class. Some features of PRIME computers, in general, and of the PRIME 750, in particular, are discussed in the following paragraphs.

Central Processor. The central processor uses 48-bit instruction words and includes hardware implemented multiply and divide instructions in both single and double precision. The advertised floating point instruction times for the 750 are somewhat slower than the VAX-11/780 times but should provide competitive performance. One very important feature of the CPU is the ease with which the machine can be upgraded. For example, to upgrade from a PRIME 350 to a 400 involves replacing one board. This will be important in the future if an increase in computing power becomes necessary after PRIME announces a machine which is faster than the 750.

Memory. The PRIME 750 memory system consists of up to 8M bytes of MOS, 540 ns cycle time, error correcting memory. This is augmented by a 16K bytes, 80 ns cache memory which reduces the effective cycle time to 149 ns. The memory is built using 16K chips with 256K bytes on a board. Therefore, 1M bytes of memory occupies just four vertical inches of cabinet space.

TABLE 2. SUMMARY OF CANDIDATE SYSTEMS

	PRIME 750	VAX-11/780	IBM 4341 †
Memory:			
Capacity/Interleaving	8M bytes/2X	8M bytes/2X	4M bytes/*
Addressability (virtual)	32M bytes	2.2G bytes	16M bytes
Cycle Time/Effective (w/cache)	540 ns/149 ns	600 ns/280 ns	*/225 ns
Bandwidth	8M bytes	13.3M bytes	9M bytes
Cache Capacity/Cycle Time	16K bytes/80 ns	8K bytes/200 ns	8K bytes/150 ns
CPU:			
Floating Point Timing (us)			
Single Precision +	0.9	0.8	1.4
* /	2.1	1.2	3.8
	*	4.2	6.3
Double Precision +	1.1	1.4	1.4
* /	3.2	3.4	5.4
	6.5	8.8	11.0
Extd Precision +	**	**	3.8
* /	**	**	15.7
	**	**	**
F.P. Accuracy			
(single/double/extended) in digits	6/13/**	6/13/**	6/13/33
F.P. Range	$10^{+38, -38}$	$10^{+38, -38}$	$10^{+75, -79}$

* Value not available.

** Feature not available.

† The IBM 4341 is a mainframe which is treated as a minicomputer for the purpose of this report.

PRIMOS IV is the operating system which supports the PRIME virtual memory system. The PRIME 750 offers 32M bytes of virtual address space to each user and 256M bytes to be shared by all users.

Input/Output. The PRIME 750 supports a maximum transfer rate of 8M bytes per second. Up to 2.4 billion bytes of disk storage can be installed on-line, including disks ranging from a 300K byte dual floppy or 3M byte cartridge to a 300M byte removable disk pack.

Software. The PRIME 750 software includes the PRIMOS IV operating system; a BASIC Interpreter; FORTRAN, COBOL, RPG II, and BASIC/VM compilers; and a macro-assembler. The operating system is written in FORTRAN, and the source code is available at no additional charge. PRIME also offers networking software which allows interconnection of up to three PRIME computers with full file sharing

capabilities. The remote job entry (RJE) emulator software consists of a single executive and HASP, 200 UT, 2780, and ICL 7020 emulator modules. The operating system and compilers support shared reentrant code, and the BASIC/VM compiler is itself reentrant. Finally, all applications software written for a PRIME computer is upward and downward compatible throughout the entire line of PRIME computers.

DEC. Digital Equipment Corporation is the leading manufacturer of minicomputer systems. The VAX-11/780 was announced in 1977, and is the only 32-bit minicomputer offered by DEC.

Central Processor. The VAX uses a microcoded 32-bit instruction set and can handle two classes of instructions. In native mode, the VAX executes instructions designed for the machine; and in PDP-11 mode, the PDP-11 instruction set is available. The systems can execute native programs and PDP-11 programs concurrently, and context switching is done in one instruction. Also, the microcode feature allows the user to implement single instructions to execute processes which would normally require many instructions to complete.

Memory. The VAX can support up to 8M bytes of MOS memory and offers 4G bytes of virtual memory. The 8K byte cache system combines with main memory to provide an effective cycle time of 280 ns. Up to 2^{31} bytes (2.2G bytes) of virtual memory is available to each user.

Input/Output. The VAX can support a memory transfer rate of 13.3M bytes per second, matching the internal bus bandwidth. This rate is made possible using buffered memory controllers. The operating system software supports the high data transfer rates by using the hardware priority levels to decrease response time, and by overlapping disk data transfer operations with seek requests.

Software. VAX/VMS is the virtual memory operating system written for the VAX-11/780. It supports real-time, interactive, and batch environments concurrently or in any combination. The operating system supports FORTRAN, COBOL, BASIC, RPG II, BLISS, and MARCO assembler programming languages. Other features offered by VAX/VMS include record management services (RMS) for general purpose file and record handling capabilities, user access procedures, resource and performance statistics, error logging, and on-line diagnostics.

IBM. The IBM 4300 series, although it is not considered a minicomputer, possesses most of the virtues of one (small size, air-cooled, low cost commercial 60-cycle power required). It will be considered a minicomputer for this report.

Central Processor. The IBM 4341 uses a microcoded instruction set. Performance tests have shown the 4341 to run up to 3.2 times faster than the IBM System/370 Model 138, and up to 3.14 times faster than a System/370 Model 148 executing the same FORTRAN job (Ref. 1). Extended precision floating point arithmetic is standard in the 4341, and offers up to 33 decimal digits of precision.

Memory. The 4341 uses 8K bytes of cache memory, 2M or 4M bytes of processor storage (physical memory), and 16M bytes of virtual storage. Memory is accessed in eight byte blocks. The effective memory cycle time typically ranges from 150 to 300 ns.

Input/Output. The processor can accommodate three or six integrated data channels, and can realize a data transfer rate of 9M bytes per second. This is accomplished with the standard byte multiplexer and two block multiplexer channels plus the three optional block multiplexer channels. Each channel can support up to eight external device controllers.

Software. Any program written for an IBM System/370 will run on the 4341 provided it is not time-dependent; is not dependent on configuration-unique features such as memory size, specific I/O equipment, or optional features being present when not included in the configuration; is not dependent on system facilities such as interrupts or certain operation codes being absent when they are included in the configuration; and is not dependent on results or functions which IBM specifies to be model-dependent or unpredictable. Any program written for a System/360 will run on the 4341 provided the above constraints are satisfied and provided it does not depend on functions which differ between the System/360 and System/370 (Ref. 2).

Operating systems which support the IBM 4341 include DOS/VSE, VM/370 Release 6, and OS/VSI Release 7. The latter two operating systems provide new functions and complete support for the 4341.

An extensive selection of compilers, communications software, and operating system support software is available for the 4341. Compilers include FORTRAN, COBOL, PL/I, and RPG II. Other software includes Sort/Merge, various communications access methods, and data base management software.

-
1. *IBM 4341 Processor Facts Folder*, G520-3387-0, IBM Corp., White Plains, New York, 1979.
 2. *A Datapro Feature Report, IBM 4300 Series*, Datapro Research Corp., Delran, New Jersey, 1978.

ARRAY PROCESSORS

In some cases, peripheral devices are available which could enhance some aspect of system performance, but which are not manufactured nor supplied by the system vendor. If such a device is considered essential, then any candidate mini-computer system must be capable of interfacing with and supporting it. One example is an array processor of the type manufactured by Floating Point Systems, Inc. Due to the relatively slow floating point processor speeds of the current minicomputers (compared with the CYBER 176), an array processor could be required to provide sufficient CP power.

The array processor is a new species of computer which is feasible because of the development of a complementary machine architecture and instruction set. Optimized tradeoffs in timing characteristics permit the use of medium and large-scale integrated circuits. Thus, off-the-shelf components are available to build a device that is capable of more than 10 to 100 times the floating point speeds of most standard computer products which are currently available (Ref. 3).

A large number of general purpose array processors are available at this time, and new or improved versions are being introduced which provide even greater speed. Following are the brief descriptions of some of these processors.

Floating Point Systems (AP-120B). The first company to introduce a general purpose, inexpensive, floating point array processor was Floating Point Systems, Inc. (FPS). The AP-120B operates in parallel with the host computer and can be interfaced to many different minicomputers and large-scale processors. It is a programmable, synchronous pipelined processor consisting of a number of fast registers, floating point adder, floating point multiplier, integer address calculator/indexer, and memory. Conversion from and to the host processor floating point format is performed on the fly to internal 38-bit format which uses a 10-bit binary exponent and 28-bit two's complement mantissa.

There is a large amount of general purpose software available for the average FORTRAN programmer. Access to the processor is accomplished through standard FORTRAN subroutine calls to a library of FPS interface routines. This library is available for each processor/operating system combination which can be interfaced to the array processor, and is responsible for handling all parameter passing and data transfers and for transmitting and initiating the array processor's

3. Davis, A. F., "Array Processor," *Industrial Research*, 1977.

programs. In addition, software is available for algorithm development and debug off-line to the array processor.

The AP-120B has a 167 ns instruction cycle time and memory fast enough to keep up. Due to the segmented functional unit architecture, a floating point add can be initiated on each cycle with a result available in two cycle times (333 ns); a floating multiply can be started each cycle with a result available in three cycle times (500 ns). Since the add and multiply can proceed concurrently, it is possible to achieve a maximum of 12 MFLOPS.

Floating Point Systems (FPS-100). The FPS-100 is a newer processor which is upward compatible with the AP-120B. Therefore, software written for the AP-120B will also run on the FPS-100. Although the FPS-100 was designed to satisfy real-time requirements, its smaller price tag and relatively high speed operation (up to 8 MFLOPS) make it a viable candidate for use in the large code processing environment.

The FPS-100 combines a priority-interrupt structure with a new resident multitasking operating system. Three internal priority levels and 15 external levels (for receipt of clock and I/O device interrupts) are available for real-time program control. In addition to a library of over 250 FORTRAN-callable math routines, a resident FORTRAN compiler is available.

Also available with the FPS-100 is a new programmable General Purpose Input/Output Processor (GPIOP) for interfacing to a wide variety of standard or custom peripheral devices. This includes the ability to interface up to 1.2G bytes of on-line data storage through 80M byte or 300M byte disk options.

Computer Design and Applications (MSP). Computer Design and Applications, Inc., (CDA) makes a microprogrammable 24-bit array processor. The Micro Signal Processor (MSP) is not nearly as flexible as the FPS processor, since all software is stored in prom and cannot be loaded dynamically as in the AP-120B. However, an assembler and simulator are provided for program development on the host, and the processor code can be referenced from a FORTRAN program. The MSP is capable of about half the speed of the AP-120B.

Computer Signal Processing (MAP 300). Computer Signal Processing, Inc. (CSPI), makes the Macro Arithmetic Processor (MAP) array processor. This 32-bit processor is user programmable and comes with an extensive library of FORTRAN callable routines which act as user interface modules on the host computer. The MAP is designed to be a powerful data acquisition, auxiliary fast I/O, and

signal processing device. It is also an arithmetic array processor. The MAP contains programmable, bidirectional interfaces between external devices and MAP memory. These devices (called scrolls) relieve the host of the I/O burden of feeding the MAP, and operate at data rates up to 36M bytes per second.

The MAP 300 contains a Central System Processor Unit (CSPU) executive processor and two arithmetic processors. The CSPU initiates processing sequences in the arithmetic processors and controls data flow in the system, including control of the I/O scrolls. It has a 16-bit fixed point arithmetic unit with a 125 ns cycle time and a sufficiently large instruction repertoire to accomplish its control functions.

Each arithmetic processor unit has completely parallel add and multiply functional units, and this parallelism exists between the processors. Therefore, two adds and two multiplies can be accomplished simultaneously. The processor performs 32-bit self-normalizing computations using an IBM format which consists of a six-hexadecimal digit mantissa, a sign bit, and a 7-bit hexadecimal exponent. Floating point operands have a dynamic range of $10^{+76, -77}$.

The MAP 300 can directly address up to 256K bytes of 32-bit memory on each of three busses. Since each bus operates independently, each of several processors can access a memory simultaneously and continuously on a cycle stealing basis. This allows for very high speed processing situations where input, output, and processing operations are overlapped. Memory is available as 500 ns 300 ns, or 170 ns cycle time MOS.

Computer Signal Processing (MAP 6400). The MPA 6400 is another recent addition to the array processor field. It interfaces to most popular 16- and 32-bit minicomputers and uses a 64-bit hexadecimal floating point number format to provide 16 decimal digits of accuracy. It operates in parallel with the host processor to provide results at speeds 10 to 1000 times faster than a minicomputer executing similar 64-bit calculations. For example, 1 second is required to calculate the product of two 100 x 100 real matrixes. Including the time required for all data fetches and stores and for program control, this computation yields an effective rate of 2 MFLOPS. More detailed information on the internal architecture of the MAP 6400 is not available at this time.

The MAP 6400 has a resident operating system and executive processor to handle task sequencing and control functions. In addition, a SNAP-II library of several hundred FORTRAN callable scientific and engineering functions is available.

CODE CONVERSION REQUIREMENTS

GENERAL

The amount of effort required to convert a particular code to run on the mini-computer system must be determined. This information together with the expected life of the code should be sufficient to decide whether or not the conversion would be justified. Any new algorithms or numerical techniques should be implemented during the conversion process. The following discussion addresses several factors which must be considered when determining conversion requirements.

ASSEMBLER CODE

Assembly language modules must be eliminated or converted to FORTRAN. If a module was written to perform a special function which does not exist on the minicomputer (call a PPU routine, attach a permanent file, request a tape), then some redesign will be required to eliminate the code. If it was written to optimize a function which could have been written in FORTRAN, then the code must be converted. In either case, assembly language modules can require a significant amount of time in the conversion process.

SEGMENTATION AND OVERLAYING

Main memory constraints could be extreme, depending upon the addressing capability of the minicomputer. At this time only three types of minis can be identified which are virtual memory machines, and which also satisfy the additional constraints that we have imposed such as 32-bit architecture and support of a fast and large capacity disk unit. These are the PRIME computers, the DEC VAX-11/780, and the IBM 4300. Others, such as the Perkin-Elmer Model 8/32 and Model 3200, and the SEL 32/75 have megabyte addressability, while most others are restricted to a 64K byte address space. Megabyte direct addressability requires the costly addition of large amounts of main memory, while 64K byte addressability results in a restrictive environment in which code conversion could be a very costly process. If one of the virtual memory machines is selected, then the problem of conforming to memory constraints could become one of simply disassembling the segment or overlay structure to run in the larger address space. In either case, memory constraints could pose some problems in conversion.

INPUT/OUTPUT

Potentially the most troublesome area in the code conversion process is that of I/O. The transportation of a large data base from one machine to another can be a major problem. Inconsistencies and conversion problems can exist in word size, character and numeric data formats, and file structures. Random access files can be particularly difficult to transport, since they must first be converted to sequential format.

ECONOMIC ANALYSIS

GENERAL

The cost of purchasing and maintaining a minicomputer system is a very important factor in determining whether or not the minicomputer approach to the large code problem is feasible and justified. Therefore, it is important that all costs and potential cost savings be identified. However, due to the subjective nature and uncertainty of costs incurred as a result of poor turnaround and losses due to lost output or improperly run jobs on the CYBER 176, these costs are not included in the analysis. Only actual computer charges are considered.

The ALFA code was chosen from those identified in the Candidate Codes section. Current costs of running this code include only actual dollar charges for CYBER 176 time. However, the total analysis includes not only a comparison of costs to run the ALFA code on the CYBER 176 and on the minicomputer, but also the estimated costs to convert the code and document it.

In addition to the actual codes, the benchmark codes should be included in the cost analysis. Due to the more strict definition of these codes, it is easier to determine the total number of floating point operations performed and the total amount of I/O involved. These numbers can be used to calculate a cost per unit of CPU and I/O. For example, the costs might be expressed in dollars per MFLOP and dollars per MWIO. Benchmark codes can be developed and run after the feasibility of the proposed minicomputer solution to the large code problem is accepted. It is highly desirable that a minicomputer of the type selected for the baseline system be used to run the benchmark codes for the cost comparison. However, if this is not possible, then the minicomputer costs would need to be estimated.

ALFA CYBER 176 COST PROJECTIONS

Table 1, shows that ALFA can require up to two CP hours per run. If we assume a modest 6 hours per week of useful run time for the ALFA code for 40 weeks per year, and if we use a nominal cost of \$800 per CP hour then we find that computer costs alone would reach \$192,000 per year for just the ALFA code. It seems clear that if other codes were included, much more substantial costs would be evident.

ALFA CODE CONVERSION AND DOCUMENTATION COSTS

ALFA and the DYNDIM preprocessor are written entirely in FORTRAN and are not overlaid or segmented. These characteristics indicate that the conversion would be minimal. However, let us assume that six person-months would be required to convert and fully document the code. Then the cost for this effort would be approximately \$30,000 in labor. Computer costs should be minimal, but again we assume an inflated figure of 2 hours of CYBER 176 time, or \$1600.

PRIME 750 COMPUTER SYSTEM

The PRIME 750 with 2M bytes of MOS, error correcting memory was chosen as the representative baseline system for this application. Appropriate mass storage, I/O peripherals, communications hardware and software, and graphics hardware is also included in the initial configuration. In addition, a nine-track tape unit is included for code and results transportability between the PRIME and CYBER machines.

Table 3 lists the hardware and software required together with the purchase and monthly maintenance costs. It can be seen that complete cost recovery can be realized in less than 3 years if just the ALFA code is considered, since the total cost of the hardware, software, 3 years maintenance, and ALFA conversion and documentation is approximately \$540,776.

TABLE 3. LIST OF COMPUTER EQUIPMENT

Item Description	Purchase Price	Monthly Maintenance
1. PRIME 750, CPU, 500K bytes of error correcting MOS memory, 24 board chassis, three 120 A power supplies, 16K bytes 80 ns cache memory, high performance 16-line asynchronous multiline controller, system console, PRIMOS operating system.	\$117,000	\$785
2. 1/2M bytes error correcting MOS memory	22,500	180
3. 1M bytes error correcting MOS memory	36,000	360
4. 300M byte disk unit with controller	42,000	300
5. 300M byte disk unit	33,000	260
6. 800/1600 bpi, 75 ips, 9-track tape drive w/controller	19,500	155
7. 600 line-per-minute printer	18,500	150
8. Versatec 200 dots-per-inch printer/plotter	15,000	130
9. Controller for Versatec	2,700	25
10. Cable for Versatec	400	NC
11. Four cables for asynchronous multiline controller	200	NC
12. 300 card-per-minute card reader	3,500	35
13. FORTRAN 77	4,150	41
14. Basic Interpreter	NC	NC
15. Two Racal-Vadic 3467P Modems (1200 Baud)	1,700	NA
16. 16-channel chassis for modems	450	NA
17. Two power supplies for modems	400	NA
18. MAP 6400 array processor with 16K program memory, 32K 64-bit data memory, and all required software and associated hardware	89,000	445
TOTAL	\$406,000	\$2,866

BENCHMARK SET

GENERAL

Programs must be written to allow the performance of various systems to be measured and compared. These programs should be written in ANSI FORTRAN and should contain no code which is dependent upon such hardware characteristics as word size. Two types of programs should be developed. One is the type which measures some performance characteristic such as floating point speed or I/O rate. We refer to this type as a quantitative benchmark. The second is the type which simulates the processing and I/O requirements of an actual code, and is called a qualitative benchmark.

QUANTITATIVE BENCHMARKS

As a minimum, three quantitative benchmark codes should be developed to measure floating point speed, floating point accuracy, and I/O performance. Floating point speed should be measured for scalar and vector add, multiply and divide operations in single and double precision. Floating point accuracy should be measured in terms of range of the exponent and number of significant digits available in the mantissa. The I/O performance benchmark should measure the number of bytes per second which can be transferred between memory and the disk.

QUALITATIVE BENCHMARKS

A qualitative benchmark measures how well the machine executes a particular code. A single performance characteristic such as I/O rate is not measured. Rather, the code is executed from start to finish and the elapsed time becomes the measure of performance for that code. Other measures can be obtained by running two or more qualitative benchmarks in a multitasking environment.

BENCHMARKING AN ARRAY PROCESSOR

If the benchmark is to be run on a minicomputer equipped with an array processor, it will be necessary to modify the CYBER 176 version of the program to take advantage of the vector processing capabilities of the minicomputer. The array processor is treated as an allocatable peripheral device; it is assigned to one task or program for the duration of execution of that program. A library of FORTRAN callable subroutines is resident in the minicomputer to serve as an interface between the calling program and the array processor. Calls to these subroutines must be inserted in appropriate places and substituted for existing loops to execute code as vector operations

CONCLUSIONS AND RECOMMENDATIONS

This study has shown that sufficiently powerful hardware is available in the minicomputer and array processor markets to provide an environment for high-speed computation for some of the large codes now running at AFWL. These codes currently require a very large share of the computer resources available to users of the AFWL scientific computer facility. If they were converted to run on a minicomputer equipped with a large amount of mass storage, high-speed computational capability, and a high degree of main memory addressability, they could be run more efficiently with a resulting decrease in the computational load on the large computers.

Three possible computers were recommended. The PRIME 750 was chosen as a baseline system, and the resulting economic analysis shows that the minicomputer solution to the large code problem is economically justified. Indeed, using only the cost of running the ALFA code shows that the minicomputer would pay for itself in less than 3 years.

It is recommended that work on this project be continued. A benchmark set should be developed to demonstrate the computing power available in the minicomputer market and the cost effectiveness of this approach. When this is complete and the results of this report justified, actions should be initiated to purchase a suitable minicomputer system to be installed at the AFWL. The system should be made available to selected users who are responsible to run those codes which are best suited to conversion to the minicomputer environment. A significant reduction in the cost of running these codes, as well as a reduction in the work backlog on the large machines, would result.

ABBREVIATIONS AND SYMBOLS

CP	Central processor; the computational unit of a computer system
ECS	Extended core storage; an extension of the high speed main memory on the CDC 6600
G	Abbreviation for 1024^3 ; 1G byte (one gigabyte) is slightly more than one billion bytes; 1,073,742,824
I/O	Input/output
K	Abbreviation for 1024 ; 1K byte (one kilobyte)
LCM	Large core memory; an extension of the high speed main memory on the CDC CYBER 176
M	Abbreviation for 1024^2 ; 1M byte (one megabyte) is slightly more than one million bytes; 1,048,576
MFLOP	Millions of floating point operations
MFLOPS	Millions of floating point operations per second
MWIO	Millions of words of input and output
ns	Nanosecond; one-billionth of a second
RJE	Remote job entry; a means of entering a job into a computer remotely over telephone lines